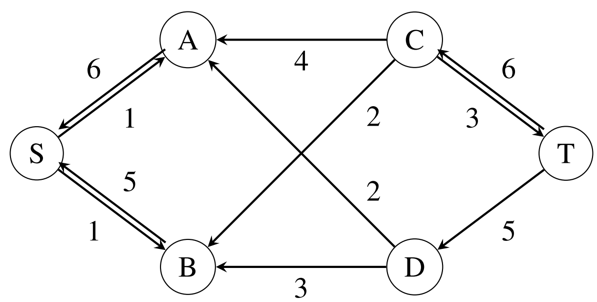
1. (20 pts.) **Network Flows.**

(a) The maximum flow is 11, with the following flows:

- 3 units on $S \to B \to D \to T$;
- 2 units on $S \to B \to C \to T$;
- 2 units on $S \to A \to D \to T$;
- 4 units on $S \to A \to C \to T$;

A minimum cut is between $\{S,A,B\}$ and $\{C,D,T\}$, with a value of 11. (The cut edges are $A \to C$, $A \to D$, $B \to D$, $B \to C$.)
Another minimum cut is between $\{S,A,B,D\}$ and $\{C,T\}$, with a value of 11. (The cut edges are $A \to C$, $B \to C$ and $D \to T$)

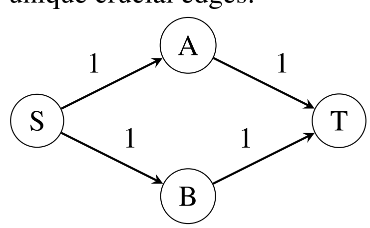
(b) The residual graph G_f is as follows:



$\{S,A,B\}$ are reachable from S, and from $\{C,T\}$ we can reach T.

(c) $A \to C$ and $B \to C$ are all crucial edges in the above graph because they are in every minimum cut of the graph. Note that increasing (A,D) or (B,D) does not change the value of the max flow, because doing so makes $[SABD, CT]$ the new min cut with value 11, so the max flow value does not change.

(d) The following graph does not have unique crucial edges:



(e) Run the usual network flow algorithm to get the residual graph G_f. In G_f, compute $\mathscr{S}$ as the vertices reachable from S, and $\mathscr{T}$ as the vertices which can reach T (by reversing edges in the residual graph), in linear time using any graph searching algorithm (BFS/DFS). An edge $u \to v$ is a crucial edge iff u is in $\mathscr{S}$ and v is in $\mathscr{T}$, and all of them can be identified in linear time.

2. (20 pts.) **Decreased Capacity.** Note that the maximum flow in the new network will be either f or $f-1$ (because changing one edge can change the capacity of the min-cut by at most 1). Now consider the residual graph of G, taking the capacity of edge (u,v) as c_{uv}. Since there is a flow of at least 1 unit going from v to t, the residual graph must have a path from t to v (each edge along which there is a flow creates a backward edge in the residual graph). Similarly, there must be a path in the residual graph from u to s, since at least 1 unit of flow reaches u from s.

Find the paths by doing a DFS from u and t, and send back 1 unit of flow through this path. This changes the flow through edge (u,v) to $c_{uv}-1$. Notice that this is a valid flow even if we replace the capacity of edge (u,v) by $c_{uv}-1$. The flow through the new graph is now $f-1$, with the edge (u,v) having capacity $c_{uv}-1$ and a flow of $c_{uv}-1$ through it. This is just an intermediate stage of the Ford-Fulkerson algorithm. If it is possible to increase the flow, then there must be an $s-t$ path in the residual graph. This can be checked by a DFS (or BFS). Since the algorithm just involves calling DFS three times, the running time is $O(|V|+|E|)$

3. (20 pts.) **Edge-Disjoint Paths.**

(a) Simply set $C(e)$ to 1 for all edges in E. The correctness of this decision is shown in parts (b) and (c).

(b) Every edge-disjoint path from s to t can carry exactly 1 unit of flow given the choice of $C(e)=1$ from part (a). Additionally, because all of the k paths are edge-disjoint, there will be no conflicts if each path is assigned its maximum amount of flow. So, we can guarantee 1 unit from each of the k paths, for a total flow value of at least k.

(c) Assume Ford-Fulkerson has found a flow of value k. We will show that there must be at least k disjoint paths in G by using induction on the flow value k.
Base Case: Let $k=0$; the discovered flow function pushes no flow. Trivially, there are at least 0 edge-disjoint paths from s to t. Inductive Hypothesis: Assume that a flow of value $k-1$ implies that there must be at least $k-1$ edge-disjoint paths for every positive integer k. Inductive Step: Let k be any positive integer. Because k is positive, some flow must be pushed through some s-t path p. By the construction of our capacity function and the assumption that all flow values are integers, we know that the flow along p must be exactly 1. Note that we can ignore the possibility that there are cycles in this s-t path because Ford-Fulkerson does not produce cycles in its flow assignments. Knowing all of this, we can count p as one of our edge-disjoint paths. To find the other $k-1$ paths, we create a new flow f' by setting the flow along every edge of p to 0. This will remove the 1 unit of flow that was being pushed through p, meaning f' has a total flow value of $k-1$. At this point, the inductive hypothesis assures us that because f' is a flow with value $k-1$, there must be at least $k-1$ edge-disjoint paths in G. Adding p to that count gives us our final result that there are at least k edge-disjoint paths in G.

(d) The running time of Ford-Fulkerson discussed in class is $O(C(|V|+|E|))$, where C is the sum of the edge capacities out of s. In general, s can never have more than $O(|V|)$ outgoing edges, one to every other node, and here, every edge has capacity 1. Therefore, $C=O(|V|)$ and the total running time is $O(|V|^2+|V||E|)$.

4. (20 pts.) **Ford-Fulkerson with Irrational Numbers.**

(a) You can pass a units of flow from the path $s-V_1-t$, and a units from $s-V_4-t$. Also, we can pass 1 unit of flow from the path $s-V_2-V_3-t$, so the total amount of flow you can pass is $2a+1$

(b) The step of the Ford-Fulkerson algorithm are shown in the following table.
The final row of this table shows that the capacities of the residual edges are of the requested form after augmenting through path p_3.

(c) After step 1 as well as after step 5, the residual capacities of edges V_2-V_1, V_4-V_3 and V_2-V_3 are in the form r^n, r^{n+1} and 0, respectively, for some $n\in\mathbb{N}$. This means that we can use augmenting paths p_1, p_2, p_1 and p_3 infinitely many times and residual capacities of these edges will always be in the same form. The total flow in the network after step 5 is $1+2(r^1+r^2)$. If we continue to use augmenting paths as above, the total flow converges to $1+2\sum_{i=1}^{\infty}r^i=1+\sum_{i=1}^{\infty}2r^i=1+\sum_{i=1}^{\infty}(2r)r^{i-1}$.

Step	Augmenting path	Sent flow	Edge $V_2 - V_1$ in Residual graph	Edge $V_4 - V_3$ in Residual graph	Edge $V_2 - V_3$ in Residual graph
0			$r^0 = 1$	r	1
1	p_0	1	r^0	r^1	0
2	p_1	r^1	r^2	0	r^1
3	p_2	r^1	r^2	r^1	0
4	p_1	r^2	0	r^3	r^2
5	p_3	r^2	r^2	r^3	0

Table 1: Ford-Fulkerson iterations

This is a geometric series with $r < 1$, so we can use the convergence formula to say that it is equivalent to $1 + \frac{2r}{1-r}$. To arrive at the final simplified form, we can use several facts stemming from $r = \phi - 1$, or one less than the golden ratio. It is known that $\phi - 1 = \frac{1}{\phi}$ and $\phi^2 = \phi + 1$. This is helpful because it can then be shown that $\frac{1}{1-r} = r + 2$, which gives us $1 + 2r(r+2) = 1 + 4r + 2r^2$. It can also be shown that $r^2 = 1 - r$, which gives us our final value of $3 + 2r$.

5. (20 pts.) **Matching Rooks.**

(a) Let $r_1, \ldots, r_m$, and $c_1, \ldots, c_n$ to be the vertices in G, representing the m rows, and n columns of chessboard. Also, let s and t be two other nodes to act as the source and sink in the flow network. First, we define the edges of G. Connect vertex s to all vertices in $r_1, \ldots, r_m$ (this gives us m edges), and their capacities to one. Then, for all $1 \le i \le m$, and $1 \le j \le n$, create an edge between r_i and c_j, and set the capacity to zero if the square at row i and column j^{th} of chessboard is unusable, otherwise set the capacity to one. Finally, connect all $c_1, \ldots, c_n$ to t, and set their capacities to one. Now we can solve this network using any max flow algorithm. Note that a path $s \to r_i \to c_j \to t$ carrying 1 unit of flow corresponds to putting a rook in the square at row i and column j of the chessboard. In other words, we can solve the problem by examining the max flow and placing a rook on square $(i,\ j)$ if the flow between r_i and c_j is 1. Also observe that the incoming flow for each of the row vertices $(r_1, \ldots, r_m)$ is at most one, and the outgoing flow for each of the column vertices $(c_1, \ldots, c_n)$ is at most one, this means that given an integer flow, for any i, r_i is connected to at most one of the column vertices (by connection here we mean an edge with flow of 1), and for any j, c_j is connected to at most one of the row vertices. This property ensures that there is at most one rook in each row and in each column, hence the max flow will give us a valid rook placement. As a result, to solve the matching rooks problem, it suffices to find the max flow of the given graph.

(b) Using the flow network we discussed in part (a), we run the Ford-Fulkerson algorithm until we reach a maximum flow. Note that the value of the maximum flow is the maximum number of rooks we can place on the chessboard. Since we can place at most one rook in any row or column, this number is at most $\min(m,n)$. Finding each augmenting path takes $O(|V| + |E|) = O(mn)$ since there are at most mn edges and $m + n + 2$ vertices in the graph. So, the total running time of Ford-Fulkerson would be $O(f_{\max}(|V| + |E|)) = O(\min(m,n)(n+m+2) + \min(m,n)mn) = O(\min(m,n)mn)$.

Rubric:

Problem 1, 20 pts

(a) total 6 pts
- 1pt for every path
- 2pts for the minimum cut

(b) total 4 pts
- 2pt for residual graph
- 1pt for reachable vertices from S
- 1pt for vertices that can reach T

(c) 3pt for $A \to B$ and 1pts for $B \to C$

(d) 3pts for any correct graph that has no crucial

(e) 4pts for correct algorithm

Problem 2, 20 pts

- 10 pts for backward edges in the residual graph and decreasing the proper edge's flow values by 1.
- 10 pts for running 1 step of Ford Fulkerson (looking for an s-t path and increasing the flow by 1 if found).

Problem 3, 20 pts

1. 3 pts for $C(e) = 1$ for all edges.
2. 5 pts for explaining that each disjoint path can carry 1 unit of flow.
3. 7 pts for an inductive proof (base case, hypothesis, inductive step) or any other valid proof.
4. 5 pts. 2 for general FF running time, 3 for simplifying with $C = O(|V|)$

Problem 4, 20 pts

(a) 6 pts for a correct maximum flow.

(b) 7 pts for showing the pattern in some way

(c) 7 pts for giving the FF max flow as a function of r and simplifying (be lenient many students will struggle with the algebra and may argue based on computing some values with a calculator).

Problem 5, 20 pts

(a) 14 pts: 7 pts for defining edges for flow network. 7 pts for explaining how the flow network can be used to solve the matching rooks problem.

(b) 6 pts: 2 pts for using augmentation path or Ford–Fulkerson method to solve the max flow network. 4 pts for showing the running time. For running time, mn^2, or m^2n are also acceptable.